Permasalahan Lintasan Terpanjang pada Graf Berbobot dan Aplikasinya dalam Kehidupan Manusia

Alfonsus Raditya Arsadjaja (13514088)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
alfonsusradityaarsadjaja@students.itb.ac.id

***Abstrak*—Pada teori graf, terdapat banyak sekali permasalahan yang berguna bagi kehidupan manusia. Diantara permasalahan-permasalahan tersebut, masih ada yang belum dapat diselesaikan secara cepat, salah satunya adalah Permasalahan lintasan terpanjang (*Longest Path Problem*) pada graf berbobot (*weighted graph*). Makalah ini akan membahas mengenai apa itu permasalahan lintasan terpanjang, permasalahan tersebut dalam jenis-jenis graf yang berbeda, algoritma untuk menyelesaikannya, serta aplikasinya dalam kehidupan manusia.**

***Kata Kunci*—Asiklik, Berbobot, Graf, Hamilton, Kompleksitas Algoritma.**

I. PENDAHULUAN

Permasalahan lintasan terpanjang (*Longest path problem*) sudah menjadi permasalahan yang sangat umum dalam teori graf. Kebalikan dari permasalahan jalan terpendek (*shortest path problem*), *longest path problem* adalah problem yang mencari jalan (*path*) terpanjang dari suatu simpul ke simpul lain dengan jarak sepanjang mungkin, dengan setiap simpul paling banyak dilewati satu kali (simple path). Permaslalahan ini merupakan penurunan dari lintasan Hamilton, hanya untuk kasus ini, ditambahkan syarat yaitu lintasan tersebut harus memiliki panjang maksimum dari semua kemungkinan lintasan yang ada.

Sampai sekarang ini, masih belum ditemukan algoritma yang optimal untuk mendapatkan lintasan terpanjang dalam suatu graf secara general. Algoritma yang paling optimal sampai saat ini memiliki kompleksitas eksponensial. Akan tetapi, untuk beberapa kasus khusus, sudah terdapat algoritma yang dapat bekerja secara polynomial dan dapat diaplikasikan ke beberapa hal terpenting dalam kehidupan manusia.

Permasalahan ini banyak sekali dipakai untuk barang-barang elektronik, penggunaan ruang secara efektif, dan job scheduling; semua akan dibahas pada akhir makalah ini.

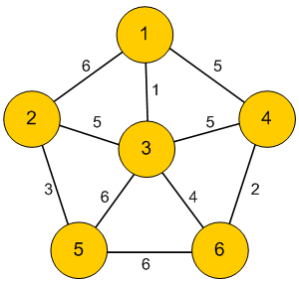
II. LANDASAN TEORI

Teori yang mendasar dalam pembahasan mengenai *Longest path problem* adalah teori graf (terutama graf berbobot), teori Hamilton, dan teori kompleksitas algoritma.

2.1. Graf

Pada dasarnya, graf merupakan kumpulan dari simpul-simpul yang saling terhubung. Graf G didefinisikan sebagai pasangan himpunan (V,E) dimana V merupakan himpunan tidak kosong (minimal ada 1 buah) dari simpul-simpul atau biasanya disebut dengan *vertices* atau *node*. Sedangkan E merupakan himpunan yang mungkin kosong dari sisi sisi yang menghubungkan sepasang simpul. Notasi untuk graf dituliskan dengan **G = (V,E)**.

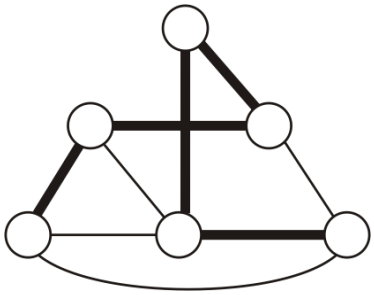
Graf bisa saja berbobot atau tidak berbobot, berarah atau tidak berarah, dan simpel atau kompleks. Pada makalah ini, graf yang dipakai adalah graf berbobot dan simpel.



Gambar 2.1 Graf
Sumber :
http://www.xatlantis.ch/examples/graph_example.html diakses 9 Desember 2015 pukul 22:11 WIB

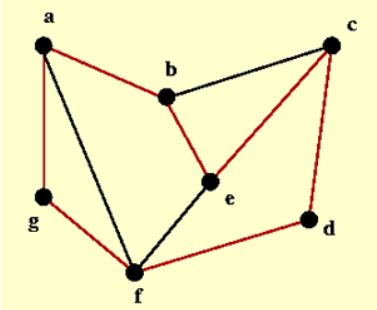
2.2. Lintasan dan Sirkuit Hamilton

Lintasan Hamilton (*Hamiltonian path*) merupakan lintasan dalam suatu graf terhubung tak berarah G dimana semua simpul dalam graf tersebut dapat dikunjungi tanpa harus mengunjungi simpul yang sama lebih dari satu kali.



Gambar 2.2 Lintasan Hamilton
Sumber :
http://www.cs.loyola.edu/~jglenn/Contest/2005/hamiltonian_path.html diakses 9 Desember 2015 pukul 23:04 WIB

Sirkuit Hamilton (*Hamiltonian cycle*) merupakan lintasan Hamilton dalam suatu graf terhubung tak berarah G, dimana lintasan tersebut berakhir pada simpul awal lintasan tersebut.



Gambar 2.3 Sirkuit Hamilton
Sumber :
http://fjw.hct.ac.ae/student_info/foundations/lsm1123/sessions/geometry/50hamiltoneancycles.html diakses 9 Desember 2015 pukul 22:44 WIB

2.3. Kompleksitas Algoritma

Kompleksitas algoritma merupakan teori yang berbicara secara luas mengenai kemangkusan (efisiensi) algoritma. Terdapat 2 macam kompleksitas, yaitu kompleksitas waktu, dan kompleksitas ruang (memori). Kompleksitas waktu diukur dari jumlah tahapan komputasi yang dibutuhkan dengan fungsi *input* sebanyak n, sedangkan kompleksitas ruang dihitung berdasarkan banyaknya memori yang dibutuhkan dengan fungsi *input* sebanyak n.

III. Permasalahan Lintasan Terpanjang dalam Berbagai Jenis Graf

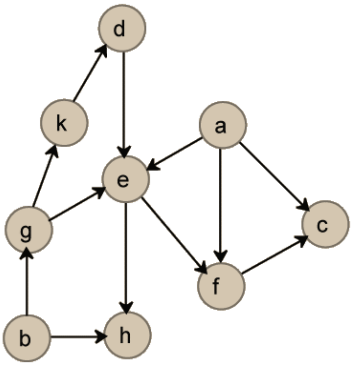
3.1. Penjelasan umum

Terdapat banyak sekali jenis graf yang dipakai dalam permasalahan lintasan terpanjang. Dikarenakan satu jenis graf berbeda dengan jenis graf yang lain, maka cara pendekatan, maupun cara penyelesaiannya pun berbeda-beda untuk setiap jenis graf. Dalam bab ini akan dijelaskan jenis-jenis graf yang dipakai, dan cara penyelesaiannya untuk masing-masing graf.

3.2. Jenis-jenis graf yang diaplikasikan

A. Graf Asiklik Berarah (Directed Acyclic Graph / DAG)

Graf G dikatakan Graf asiklik berarah apabila graf tersebut adalah graf berarah, dan tidak memiliki siklus (*cycle*) dalam graf tersebut.



Gambar 3.1 Graf asiklik berarah
Sumber : http://ramos.elo.utfsm.cl/ diakses 9 Desember 2015 pukul 01:33 WIB

Permasalahan lintasan terpanjang pada graf jenis ini memiliki properti khusus. Graf ini tidak memiliki siklus (*cycle*) sehingga dapat diselesaikan dengan pembuktian dengan memakai ide dari *shortest path problem*. Ide pembuktiannya adalah sebagai berikut:

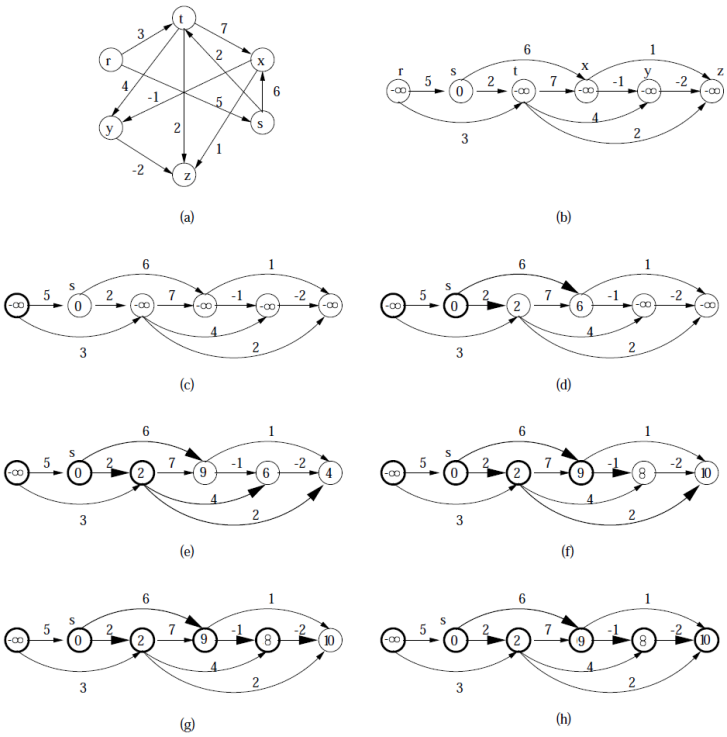
Misalkan terdapat Graf berbobot G yang akan dicari lintasan terpanjang pada graf tersebut. Misalkan juga Graf –G yang sama seperti graf G, hanya semua bobotnya menjadi negatif.

Jika bisa didapatkan lintasan terpendek pada graf -G, maka juga bisa didapatkan lintasan terpanjang pada graf G. Graf –G tidak dapat dicari lintasan terpendeknya jika terdapat siklus negatif (*negative cycle*).

Akan tetapi, karena graf G merupakan graf yang tidak mempunyai siklus, graf –G juga tidak akan mempunyai siklus. Karena itu, lintasan terpanjang pada graf G bisa dicari dengan menggunakan lintasan terpendek (total bobot terkecil) pada graf –G dengan algoritma traversal dan topological sort.

Topological sort pada graf G ini berguna untuk mencari urutan yang mungkin pada graf G, untuk mendapatkan semua kemungkinan simpul awal. Algoritma traversal pada hal ini berguna untuk mencari lintasan dengan panjang minimum pada graf G setelah dilakukan topological sort. Algoritmanya mempunyai urutan sebagai berikut:

1. Lakukan topological sort pada graf tersebut untuk mendapatkan urutan simpul-simpul pada DAG (*Directed Acyclic Graph* / Graf asiklik berarah) tersebut.



Gambar 3.2 Proses pencarian longest path pada Graf asliklik berarah
Sumber : http://www.geeksforgeeks.org/find-longest-path-directed-acyclic-graph/
diakses 10 Desember 2015 pukul 02:09 WIB

2. Untuk setiap urutan simpul v pada graf tersebut, hitunglah panjang lintasan antara tetangga dari simpul v yang memiliki sisi masuk ke v, dan maksimumkan semua jaraknya untuk menjadi jarak maksimum ke titik v. Jika tidak ada tetangga yang memiliki sisi masuk ke v, maka jarak maksimumnya adalah 0.

Setelah selesai memproses semua simpul, lakukan backtracking dari simpul paling belakang, untuk mendapatkan simpul-simpul mana sajakah yang terdapat pada lintasan yang terpanjang. Cara *backtrack*nya adalah dengan melakukan algoritma dfs dari belakang.

Anggap simpul sekarang adalah simpul v. Untuk semua tetangga yang masuk ke v, cek apakah jarak maksimum dari tetangga v ditambah dengan panjang sisi adalah jarak maksimum ke v. Jika iya, maka tetangga v ditambahkan ke daftar simpul yang termasuk pada *path* yang dimaksud, dan v sekarang menjadi tetangga v tersebut (v = prec(v) dimana prec(v) adalah predesesor v). Lakukan hal ini sampai v adalah simpul awal graf tersebut. Setelah selesai, maka didapatkan daftar simpul pada lintasan terpanjang yang dimaksud, beserta panjang lintasan tersebut.

Lintasan terpanjang tidak harus mulai dari simpul dengan banyak sisi masuk adalah 0. Sebagai contoh, untuk gambar 3.2 diatas, simpul awal adalah s (walaupun saat dilakukan *toposort*, simpul paling awal graf tersebut adalah r). Untuk kasus seperti ini, untuk langkah 2 diatas, pada awal instruksinya ditambahkan instruksi berikut:

> "inisialisasi jarak ke semua simpul menjadi nilai negatif infinity (dapat direpresentasikan dengan nilai negatif sangat besar yang tidak mungkin dicapai dari perhitungan jarak), kecuali simpul awal yang menjadi titik awal, diinisialisasi dengan 0".

Serta instruksi "Jika tidak ada tetangga yang memiliki sisi masuk ke v, maka jarak maksimumnya adalah 0." diubah menjadi :

> "Jika tidak ada tetangga yang memiliki sisi masuk ke v, maka jarak maksimum ke v tetap (negatif infinity)."

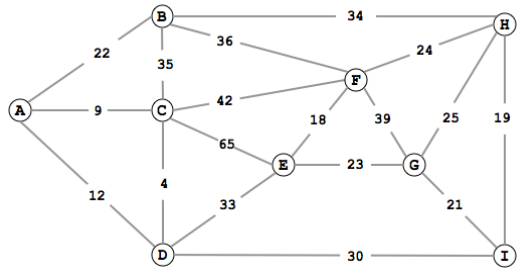
Sebagai contoh, untuk gambar 3.2 diatas, lintasan yang dimaksud adalah simpul s-t-x-z dengan panjang lintasan adalah 10.

Kompleksitas algoritma untuk melakukan topological sorting adalah O(V + E), O(V + E) untuk mencari lintasan terpanjang, dan O(V) untuk backtracking. Sehingga total kompleksitas untuk algoritma ini adalah O(V + E).

B. Graf pada umumnya

Graf jenis ini merupakan graf yang bersifat universal, dapat mencakup semua jenis graf yang lain, seperti graf planar, graf berarah, graf lengkap, dan yang lainnya. Pada graf ini, graf tidak berbobot juga dapat diaplikasikan pada graf ini, dengan mengasumsikan bobot setiap sisi adalah 1.

Pada makalah ini, akan dijelaskan algoritma pada graf yang lebih general, yaitu pada graf berbobot.



Gambar 3.3 Graf pada umumnya
Sumber : http://stackoverflow.com/questions/20807286/minimum-sum-weight-of-connecting-3-vertices-in-an-undirected-weighted-graph-wi diakses 9 Desember 2015 pukul 02:52 WIB

Graf ini bersifat universal, maka penyelesaiannya pun harus yang bersifat universal. Sampai sekarang ini, belum ada algoritma yang dapat menyelesaikan permasalahan untuk graf ini secara cepat; dengan kata lain, algoritma yang dipakai bertumbuh secara eksponensial seiring dengan makin besarnya input (n).

Algoritma terbaik saat ini yang dapat mencari lintasan dengan panjang maksimum pada graf umum, memiliki kompleksitas yang sangat besar, walaupun dapat diselesaikan secara dp (*dynamic programming*). Langkah-langkahnya adalah sebagai berikut:

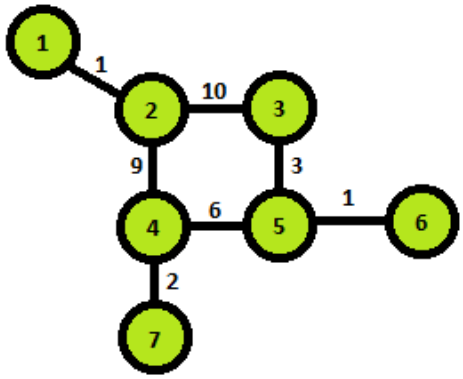
1. Melakukan traversal secara dfs (*depth-first search*). Setelah mendapatkan pohon dfs-nya (*Tremaux tree*), cari kedalaman tree tersebut. Misalkan kedalaman pohon tersebut adalah d.
2. Lakukan dekomposisi lintasan (*path decomposition*) pada pohon dfs tersebut, dengan lebar lintasan d.
3. Setelah didekomposisi, lakukan algoritma dynamic programming (dp) untuk mencari panjang lintasan maksimum pada graf tersebut.

Kompleksitas algoritma ini adalah $O(\mathrm{d}!2^{\mathrm{d}}n)$ dimana n adalah banyaknya simpul pada graf tersebut. Kompleksitas ini sangat besar, dan tidak praktikal jika panjang banyak sisi dan simpulnya lebih dari 10. Karena itu, ada beberapa aproksimasi yang dilakukan untuk dapat mengurangi kompleksitas diatas.

Terdapat algoritma yang dapat mengaproksimasi lintasan maksimum tersebut dengan waktu polynomial (bukan eksponensial). Kompleksitas pastinya dalam perhitungan panjang lintasan ini masih diperdebatkan, dan ada beberapa teorema yang menggabungkan graf dan kompleksitas yang dipakai untuk memprediksi kompleksitas keseluruhan pencarian panjang lintasan tersebut. Aproksimasi tersebut juga memakai ide dengan memanipulasi graf G dengan memakai ide dari siklus Hamiltonian dan memakai ide dari permasalahan graf dengan algoritma eksponsial lainnya, seprti TSP (*Travelling Salesman Problem*) dan permasalahan 3-SAT (*3-Sastifiability Problem*).

C. Graf khusus lainnya

Diberikan sebuah graf terhubung berbobot tidak berarah, dimana graf tersebut hanya memiliki maksimal 1 siklus.



Gambar 3.4 Graf dengan 1 siklus
Sumber : Dokumen pribadi

Permasalahan pada graf ini dapat diselesaikan dengan tinjau 2 kasus sebagai berikut:

1. Tidak ada siklus
   Permasalahan ini dapat direduksi menjadi *longest path on a tree*, yang dapat diselesaikan dengan waktu linear (*linear complexity*).
2. Mempunyai 1 siklus
   Untuk graf yang mempunyai 1 siklus, kemungkinan lintasan terpanjang terdapat pada beberapa bagian, seperti :
   i. Pada siklus tersebut
   ii. Diantara pohon yang terdapat pada simpul dan melalui bagian dari siklus tersebut
   iii. Diantara 2 simpul yang terdapat pada satu pohon yang sama

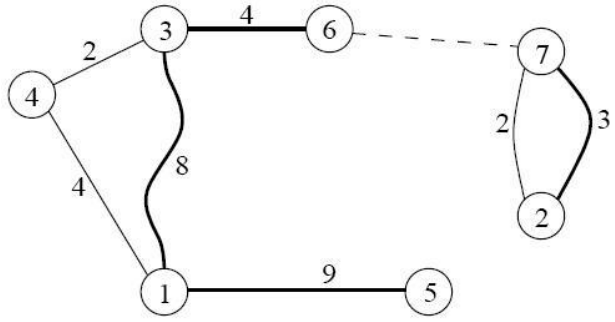
Untuk mencari algoritma yang dapat menyelesaikan sekaligus 3 kasus diatas, tentukan dahulu siklus yang terdapat pada graf tersebut. Setelah itu, tentukan simpul mana saja yang terdapat pada siklus tersebut.

Untuk setiap simpul, carilah jarak terpanjang dari simpul tersebut sampai ke simpul yang paling jauh (yang tidak melalui siklus). Jarak tersebut akan menjadi jarak sementara untuk mencari setiap kemungkinan jarak maksimum untuk 2 simpul dalam siklus tersebut. Jika pada simpul tersebut terdapat lebih dari satu sisi yang tidak termasuk dalam sisi siklus (*cycle edge*), maka cari lintasan maksimal pertama dan maksimal kedua dari simpul tersebut, lalu hasilnya dimaksimalkan dengan jumlah lintasan maksimal pertama dan maksimal kedua dari simpul tersebut.

Setelah itu, jarak tersebut dimaksimalkan dengan jarak terpanjang untuk setiap pohon (*longest path on a tree*).

Lintasan terpanjang pada graf ini merupakan hasil lintasan maksimal dari semua kemungkinan yang ada.

Salah satu permasalahan yang memakai algoritma ini adalah soal IOI 2008 – Island. Inti dari permasalahan ini adalah diberikan suatu graf dimana dari setiap simpulnya akan dibangun tepat sisi ke simpul yang lain (bebas, asal bukan diri sendiri).



Gambar 3.5 Contoh graf dalam persoalan diatas
Sumber : http://www.spoj.com/OI/problems/ISLAND/
diakses 10 Desember 2015 pukul 12:09

Tujuan dari soal ini adalah mencari total lintasan terpanjang yang tidak memiliki siklus. Graf bisa saja terpisah, akan tetapi untuk setiap graf yang terhubung, algoritma diatas dapat dipakai, karena untuk setiap graf yang terhubung, maksimal hanya terdapat satu siklus, sama seperti yang telah dijelaskan pada Bab 3.C diatas.

Total panjang lintasan adalah total dari semua panjang lintasan untuk setiap graf yang terhubung.

IV. Aplikasi pada Kehidupan Manusia

4.1 Sirkuit Terintegrasi

Menentukan lintasan penting (*critical path*) pada sirkuit terintegrasi (*integrated circuit*).

Gambar 4.1 Sirkuit Terintegrasi (*Integrated Circuit*)
Sumber : https://articlefind.wordpress.com/2012/01/
diakses 10 Desember 2015 pukul 19:06 WIB

Suatu sirkuit terintegrasi memiliki banyak komponen penyusunnya. Salah satunya adalah komponen yang bernama transistor. Setiap transistor memiliki komponen-komponen sendiri didalamnya (tidak dibahas disini). Setiap transistor pada sirkuit terintegrasi akan dialiri oleh tegangan dan arus listrik. Setiap transistor tersebut memiliki kapasitansi didalamnya. Karena tidak ada peralatan buatan manusia yang ideal, maka terdapat suatu *delay* pada setiap transistor. *Delay* ini mempengaruhi cepat lambatnya informasi diteruskan dan diproses pada komputer. Banyaknya transistor dan jarak yang berbeda-beda antar transistor juga membuat total *delay* antar transistor juga berbeda-beda.

Untuk menghitung waktu minimum informasi diproses oleh suatu IC, dapat memakai algoritma permasalahan lintasan terpanjang pada graf asiklik berarah untuk mencari *critical path* pada Sirkuit tersebut (tentunya waktu *delay* juga dihitung dengan mempertimbangkan komponen-komponen lainnya), dan untuk menentukan komponen mana saja yang sangat krusial agar waktu pemrosesan tidak semakin lambat dan dapat disusun seoptimal mungkin.

4.2 Permasalahan Perencanaan

Permasalahan Perencanaan (*Scheduling problem*) adalah problem yang berbicara mengenai permasalahan suatu proyek, yang terdiri dari lebih dari satu subproyek. Dalam suatu proyek, beberapa pekerjaan dapat dikerjakan secara bersamaan, akan tetapi ada juga pekerjaan yang mengharuskan satu atau lebih pekerjaan selesai terlebih dahulu, baru dapat dikerjakan.

Waktu minimum yang dapat diselesaikan oleh proyek tersebut dapat dicari dengan mencari *critical path* pada graf tersebut. *Critical path* pada permasalahan ini adalah lintasan pekerjaan-pekerjaan, yang apabila salah satu saja terlambat dilakukan, maka akan mempengaruhi keseluruhan waktu pengerjaan proyek tersebut.

Critical path pada permasalahan ini dapat dicari dengan algoritma *Longest Path Problem* pada Graf asiklik berarah, dan melakukan backtracking untuk mencari pekerjaan mana saja yang termasuk dalam *critical path* tersebut.

V. Kesimpulan

Algoritma pencarian lintasan terpanjang pada suatu graf secara general masih bersifat NP-hard (Nondeterministic polynomial-hard), belum ditemukan algoritma optimal yang dapat menyelesaikan permasalahan ini dengan cepat. Akan tetapi, sudah terdapat aproksimasi / pendekatan yang cukup dekat untuk mengatasi permasalahan ini.

Untuk graf dengan kasus khusus, terdapat beberapa macam, diantaranya yang memiliki maksimal 1 siklus, dan yang lebih banyak aplikasinya, yaitu graf asiklik berarah.

Aplikasi pada graf ini, mayoritas berada di bidang elektronika, salah satunya adalah mencari *critical path* untuk sebuah *integrated circuit*. Selain itu, hal ini juga berguna untuk merencanakan suatu proyek yang sangat besar, dan terintegrasi satu sama lain.

Tentunya, masih sangat banyak aplikasi yang memakai

graf asiklik berarah, dalam menentukan *critical path* pada bidang yang bersangkutan, karena memiliki kompleksitas waktu linear (*linear complexity*).

VI. Ucapan Terima Kasih

Penulis menyampaikan syukur kepada Tuhan yang Maha Kuasa, karena makalah ini dapat tersusun dengan baik. Penulis berterima kasih kepada Dr. Ir. Rinaldi Munir, M.T. dan Dra. Harlili, yang telah membimbing penulis dalam penulisan makalah ini melalui perkuliahan Matematika Diskrit. Penulis juga mengucapkan terima kasih kepada seluruh pihak yang berperan dalam penyusunan makalah ini sehingga penulis dalam menyelesaikan makalah ini tepat waktu.

Daftar Pustaka

[1] Munir, Rinaldi. 2006. *Matematika Diskrit*. Bandung:Penerbit Informatika

[2] Sedra, Adel S. & Smith, Kenneth C. 2010. *Microelectric Circuits*. Edisi ke-6. Oxford:Oxford University Press

[3] Karger, David; Motwani, Rajeev; Ramkumar, G. D. S. (1997), "On approximating the longest path in a graph", Algorithmica 18 (1): 82–98, doi:10.1007/BF02523689, MR 1432030.

[4] Gabow, Harold N.; Nie, Shuxin. 2008. "Finding long paths, cycles and circuits", International Symposium on Algorithms and Computation, Lecture Notes in Computer Science 5369, Berlin: Springer

[5] http://www.geeksforgeeks.org/find-longest-path-directed-acyclic-graph/, diakses 9 Desember 2015 pukul 01:33 WIB

[6] http://www.ioinformatics.org/locations/ioi08/contest/IOI2008Booklet1.0.pdf, diakses 10 Desember 2015 pukul 12:21 WIB

[7] https://www.quora.com/What-are-the-practical-applications-of-determining-the-longest-path-in-a-graph, diakses 8 Desember 2015 pukul 02:30 WIB

[8] http://www.tutorialspoint.com/management_concepts/critical_path_method.htm, diakses 9 Desember 2015 pukul 21:55 WIB

[9] http://stackoverflow.com/questions/19872685/relation-between-critical-path-and-the-longest-path, diakses 9 Desember 2015 pukul 22:20 WIB

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 11 Desember 2015

Alfonsus Raditya Arsadjaja
13514088